RL-TR-95-112
Final Technical Report
June 1995

KNOWLEDGE-BASED AUTOMATIC GRAPH LAYOUT

CoGenTex, Inc.

Dr. Ehud Reiter

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Rome Laboratory
Air Force Materiel Command
Griffiss Air Force Base, New York

This report has been reviewed by the Rome Laboratory Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RL-TR-95-112 has been reviewed and is approved for publication.

APPROVED: 

DOUGLAS A. WHITE
Project Engineer

FOR THE COMMANDER: 

JOHN A. GRANIERO
Chief Scientist
Command, Control & Communications Directorate

If your address has changed or if you wish to be removed from the Rome Laboratory mailing list, or if the addressee is no longer employed by your organization, please notify RL (C3CA) Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	June 1995	Final Aug 94 - Mar 95

4. TITLE AND SUBTITLE

KNOWLEDGE-BASED AUTOMATIC GRAPH LAYOUT

5. FUNDING NUMBERS

C - F30602-94-C-0281
PE - 62702F
PR - 5581
TA - 27
WU - PP

6. AUTHOR(S)

Dr. Ehud Reiter

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

CoGenTex, Inc.
Village Green, Suite 5
840 Hanshaw Road
Ithaca NY 14850-1589

8. PERFORMING ORGANIZATION REPORT NUMBER

N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Rome Laboratory (C3CA)
525 Brooks Rd
Griffiss AFB NY 13441-4505

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

RL-TR-95-112

11. SUPPLEMENTARY NOTES

Rome Laboratory Project Engineer: Douglas A. White/C3CA/(315) 330-3564

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This report describes the initial investigation of the applicationsof ideas and techniques, drawn from Natural Language Generation (NLG) experience, that can improve the quality and usefulness of diagrams produced by automatic graph layout systems. Section 1 of the report gives information on automatic graph layout and NLG, summarizes the investigation and its findings, and illustrates how some specific diagrams can by improved by the identified techniques. Section 2 summarizes the state-of-the-art. Sections 3 through 8 discuss sublanguages, pragmatics, document planning, user and task tailoring, psychological knowledge, and consistency issues. Section 9 speculates about possible document creation using "integrated multimodal theory," and Section 10 contains conclusions.

14. SUBJECT TERMS

Graph layout, Natural language generation, Automatic programming, Formal methods, Knowledge-based systems

15. NUMBER OF PAGES

52

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

1 Introduction

The goal of this project is to identify ideas and techniques which can improve the quality and usefulness of diagrams produced by Automatic Graph Layout (AGL) systems, drawing on CoGenTex's experience in Natural-Language Generation (NLG). Since 'automatic graphic generation' and 'automatic text generation' are both special cases of 'automatic document generation', we believe that substantial opportunities exist for cross-fertilization between AGL and NLG. In particular, we believe there are many concepts that have perhaps been more highly developed in the NLG community than in the AGL community, and could usefully be transitioned to AGL systems: these include *sublanguages*, *pragmatics*, and *document planning*.

The similarity that we see between techniques for "effective textual presentation of information" and "effective graphical presentation of information" also suggests that, in the longer-term, it may be possible to develop an integrated theory that can be used as the basis of all computer-to-human communication, regardless of the media. We are a long way from having such a theory, but if it could be developed, it would be of great theoretical interest, and also provide a solid foundation for attempts to build intelligent multimedia systems.

Our current project has been a small initial exploration, aimed more at identifying promising directions than at developing usable technology. We have read the AGL literature, consulted with AGL experts, experimented with state-of-the-art AGL software, and analyzed human-authored diagrams. We have also built a small demo system that shows how some of our ideas could be used (in a simple and straightforward way) to improve some of the diagrams produced by Knowledge-Based Software Assistant (KBSA) Advanced Developed Model (ADM), an advanced Computer-Aided Software Engineering (CASE) system being developed by Andersen Consulting for Rome Laboratory.

The remainder of Section 1 gives background information on AGL and NLG, summarizes our investigation and its findings, and illustrates how some specific ADM diagrams could be improved by the techniques we have identified. Section 2 summarizes previous and related work, including the current commercial state-of-the-art. The next six sections analyze the applicability of specific NLG techniques to AGL: Section 3 discusses sublanguages; Section 4 discusses pragmatics; Section 5 discusses document planning; Section 6 dis-

cusses user and task tailoring; Section 7 discusses using psychological knowledge; and Section 8 discusses consistency. Section 9 is more speculative, and discusses the possibility of creating an 'integrated multimodal theory' of document creation. Section 10 gives our concluding comments, and an Appendix describes how to run our demo system.

I very gratefully acknowledge the help of Michael White and Ted Caldwell in performing this research, creating the demo system, and writing the Final Report. Any mistakes, errors, or omissions are solely my responsibility.

1.1 Automatic Graph Layout

Automatic Graph Layout (AGL) systems automatically produce diagrams from raw computer data. In the CASE domain, for example, an AGL system might be used to automatically draw an appropriate Entity-Relation (E-R) diagram from an SQL database definition; to automatically draw a class diagram for an object-oriented system, using information obtained by analyzing C++ class definitions; or to automatically draw an annotated task/subtask tree that communicates project management data extracted from a project management database. AGL technology is starting to be incorporated in advanced CASE systems, including the ADM system being developed by Andersen Consulting for Rome Laboratory, and the ROSE system marketed by Rational.

In this project, we have focused on AGL techniques that are relevant to automatically producing E-R diagrams. This means in particular that we have focused on AGL techniques that are relevant for producing *network diagrams*, i.e., graphics which show a set of nodes (entities) and links between the nodes (relations). Other common examples of network diagrams include class diagrams, dataflow diagrams and flowcharts. We have paid less attention to AGL techniques used to produce *data graphics* such as bar charts and scatter plots, although we have read some of the (substantial) literature in this area.

The acronym AGL is often taken to mean *aesthetic* graph layout, i.e., the automatic generation of 'nice-looking' diagrams. We believe, however, that the true goal of AGL should be to generate *effective* and *understandable* diagrams, which does not always mean the same thing. That is, we believe AGL systems should be considered effective if human users find them to be a useful way of obtaining or authoring certain kinds of information; whether

the diagrams produced are pleasing to the eye or not is a secondary issue. Furthermore, effectiveness can probably only be judged in the context of a particular task; different techniques may be appropriate in different task contexts. An AGL system that works very well as an aid to a human diagram-creator, for example, may be less effective as a tool for presenting information automatically extracted from existing database definitions.

1.2 Natural Language Generation

Our background is in Natural Language Generation (NLG) technology, that is in building systems that automatically generate texts from computer data. For example, in other projects CoGenTex has developed systems that automatically generate project management reports [KMR93], weather reports[GDK94], statistical summaries [IKK+92], and job descriptions [CK94].

Perhaps for historical reasons,[1] the NLG community has often paid more explicit attention to communicative issues than the AGL community. NLG systems are often thought of as fulfilling 'task-dependent communicative goals', and evaluated in terms of how effectively they achieve their goals. The NLG community has also in many ways placed more emphasis on thinking about what criteria make a text segment effective at fulfilling a communicative goal; the AGL community, in contrast, has often tended to place primary emphasis on devising fast algorithms, and perhaps less emphasis on ensuring that what is produced by the algorithms is indeed what is wanted. Because of this, we believe that there are things that AGL system builders can learn from NLG system builders; no doubt NLG system builders can also learn from AGL system builders.

1.3 The Investigation

This project has been a small[2] initial study of the applicability of NLG ideas to AGL, aimed primarily at surveying the current state-of-the-art in AGL and evaluating which NLG ideas have the most promise for improving the

[1] The NLG community has been heavily influenced by philosophical ideas about communication, such as speech act theory [Sea69]. The AGL community, in contrast, has largely evolved out of work by algorithm and complexity theorists on graph-related algorithms.

[2] About 4 person-months.

state-of-the-art. Our goal has been more to identify promising directions than to develop fieldable technology. In particular, we have during this project

- read numerous academic papers about AGL;
- consulted with several AGL experts from universities and commercial laboratories;
- experimented with several existing AGL systems, including CLEAR Software's allClear system, Tom Sawyer's Graph Layout Toolkit [Tom94], AT&T's dotty system [NK94], the KBSA Concept Demo [DMBS92], and KBSA Advanced Development Model (ADM);[3]
- analyzed a variety of human-produced diagrams.

1.4 Summary of Findings

In summary, our conclusions about the applicability of NLG ideas to AGL are as follows:

- Graphical languages, like textual languages, are strongly affected by the *sublanguage* phenomena [GK86]. In other words, in graphics, as in text, different 'genres' have evolved, and graphics-generation systems, like text-generation systems, need to respect the genre conventions that their users expect to see obeyed. This means that it may be inappropriate to attempt to build 'universal' AGL systems, as many people in the field are attempting to do; it may be more appropriate to build highly parametrizable systems that can be customized for many different sublanguages.
- Human users attach importance to *pragmatic* phenomena such as node alignment and proximity; they do not simply look at what nodes are

[3] AllClear is a flowcharting tool that has AGL capabilities. Tom Sawyer's Graph Layout Toolkit is the only commercially available toolkit for laying out network diagrams that we are aware of. AT&T's dotty system was recommended to us as an advanced AGL tool for network diagrams that is being used in-house by a major company. The KBSA Concept Demo and Advanced Development Model are research prototypes of an advanced CASE system, and include AGL components which can produce a variety of CASE diagrams, including E-Rs.

connected to what other nodes. Current AGL systems can and should be modified to use such pragmatic features to convey information about (in particular) functional grouping and the relative importance/salience of nodes. However, at least in the immediate future, it probably will be difficult for AGL systems to detect when a diagram accidentally conveys unwanted and incorrect pragmatic inferences, i.e., determine whether a diagram is *free of false implicatures* [MR90].

- Textual documents consist of numerous paragraphs (sections, chapters, etc.); it would be inappropriate in most cases to produce a single large paragraph that goes on for several pages. In many cases, it is also useful for AGL systems to use *document planning* techniques to split up large diagrams into a structured set of smaller diagrams, perhaps connected by hypertext (hypergraphics) links.

- At a high level (e.g., diagram authoring vs. diagram browsing), the *user's task* should be taken into consideration when designing an AGL system. Automatically tailoring diagrams according to a detailed model of the user and his task (e.g., the user is an experienced analyst who is trying to validate a proposed database schema) would be desirable, but is likely to require a substantial amount of domain knowledge, which may not always be available.

- Basing AGL rules on *psychological knowledge* of the human user is a laudable long-term goal, but it will be hard to do this in the short term, given the current state of psychological knowledge.

- Diagram *consistency* is very important, but it is not clear what ideas we as NLG experts can offer to achieve this, except enforcement of a strong sublanguage. This is partially due to the fact that differences in human visual and linguistic perception may mean that different kinds of consistency are important for language and for graphics.

We can summarize many of the above points by saying that AGL systems should take into account *contextual information* as well as the actual nodes and edges that need to be drawn. This contextual information should in principle include the target sublanguage, pragmatic information about grouping and salience, the user's task, and the user's psychological limitations; we would not be surprised if subsequent research adds other factors to this list.

1.5 An Example

Figures 1, 2, 3, 4, and 5 illustrate some of the issues we are discussing. Figure 1 is an example diagram produced by the ADM AGL system. It shows an ER-like diagram for a Hotel domain.[4] This is quite a reasonable layout, and indeed the ADM system represents the state of the art in AGL systems.

Nevertheless, if we look at this diagram from the perspective of the issues mentioned in Section 1.4, it is possible to suggest improvements. In particular, Figure 2 shows the same diagram laid out with better *pragmatics*; Figure 3 shows the same diagram laid out according to the E-R *sublanguage* given in Barker's textbook (Section 2.2); Figure 4 shows an overview of the diagram produced by *diagram planning* techniques; and Figure 5 shows the overview with one subnetwork expanded.[5]

We will now describe these diagrams in more detail. Figure 2 shows how Figure 1 would appear with better *pragmatics*. In particular

- The importance of Hotel is emphasized by placing it in the center of the diagram. This is also done to a certain extent in Figure 1; however, we believe that the pragmatic fact that Hotel is the central node of the diagram is more immediately obvious in Figure 2.

- Groupings are more evident. For example, in this diagram there are three subnetworks that only intersect in the Hotel node; they are {Hotel, CarRental, Airline}, {Hotel, Employee, Position, Salary, Training, Country, Regulations}, and {Hotel, Room, Reservation, Guest, DiscountQ, and DiscountRate}. We believe that this fact is clearer in Figure 2 than in Figure 1.

Figure 3 shows how Figure 1 would appear in the E-R *sublanguage* given in Barker's textbook (Section 2.2). For example, standard E-R symbols such

[4]This diagram was produced by running the `agl-demo` program on the file `agltest.lhotel`, and asking the system to layout the diagram as a whole (instead of incrementally).

[5]Figure 2 was created by manually modifying Figure 1 using ADM's diagram editing facilities. Figure 3 was manually created with a general drawing package (Superpaint). The diagram overview shown in Figures 4 and 5 was computed by our demo system, and drawn by the ADM `agl-demo` program.

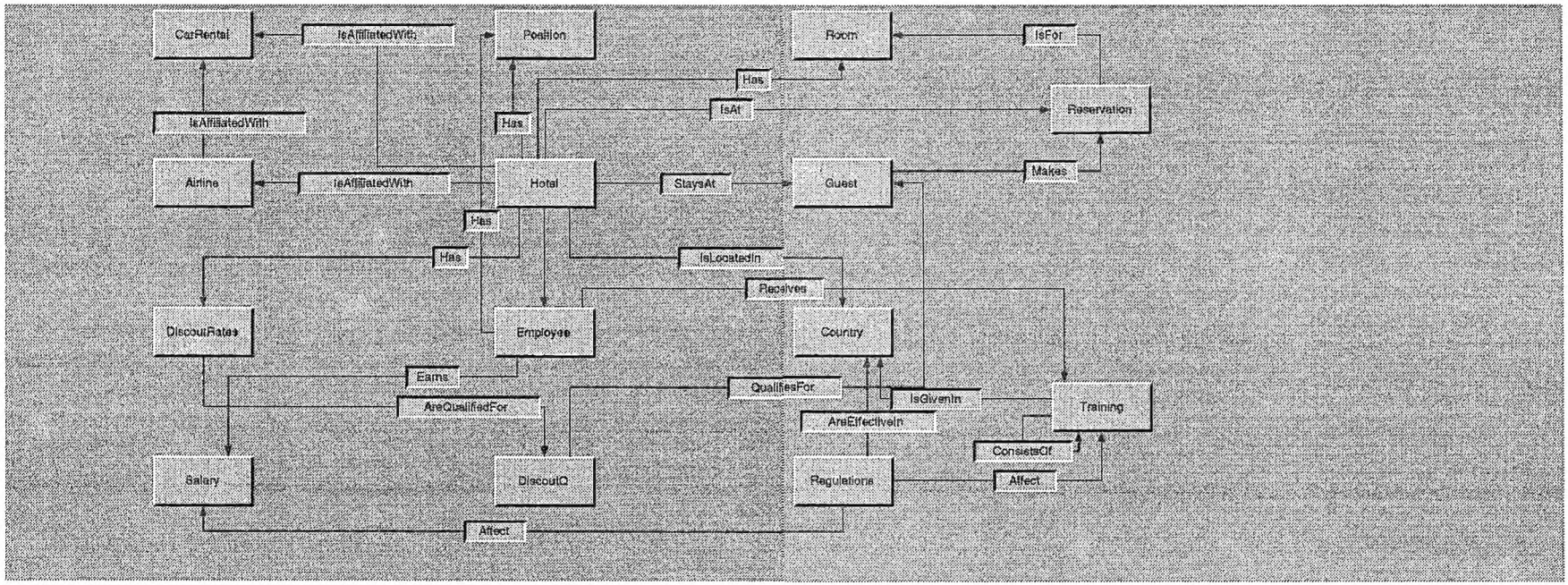

Figure 1: Hotel Diagram as Displayed by ADM

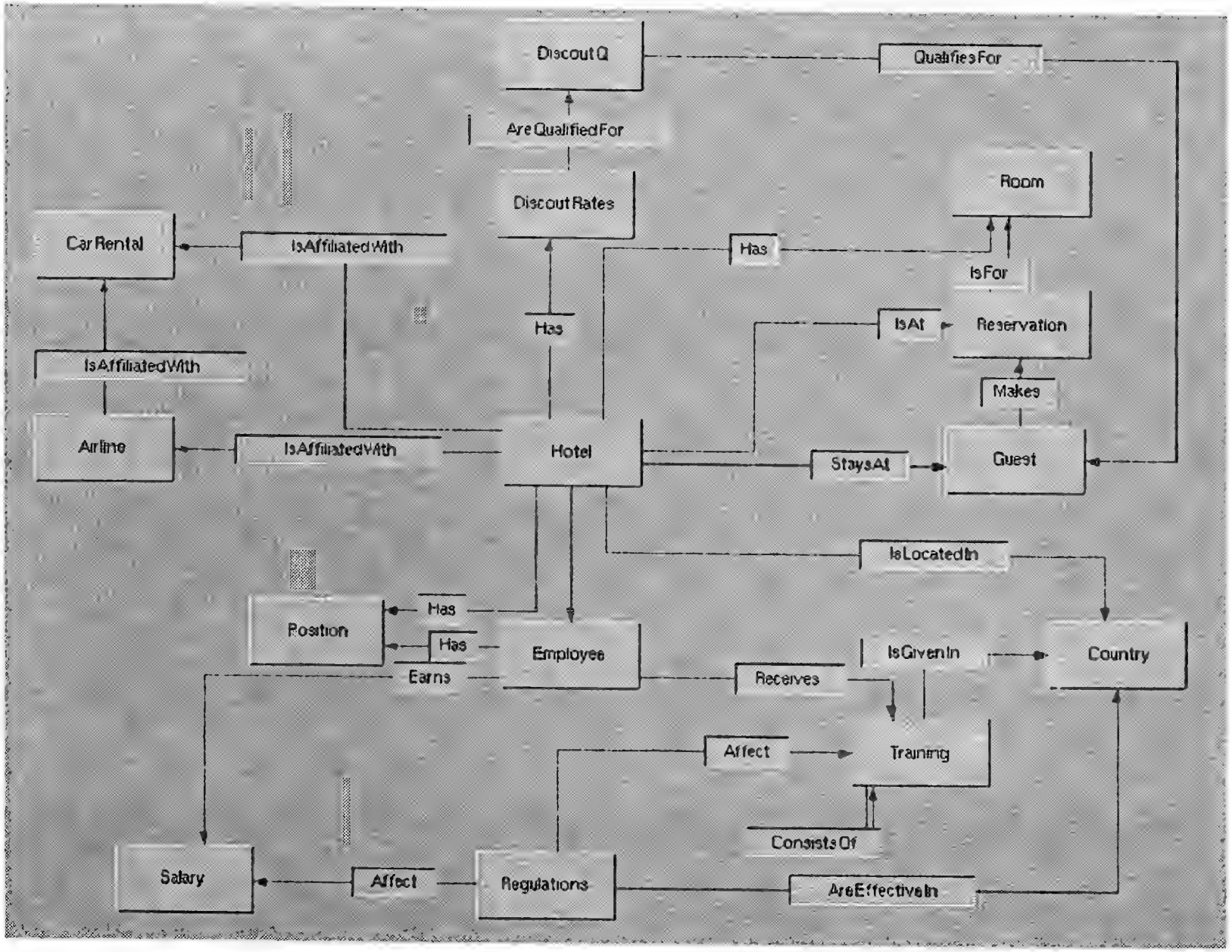

Figure 2: Hotel Diagram with Better Pragmatics

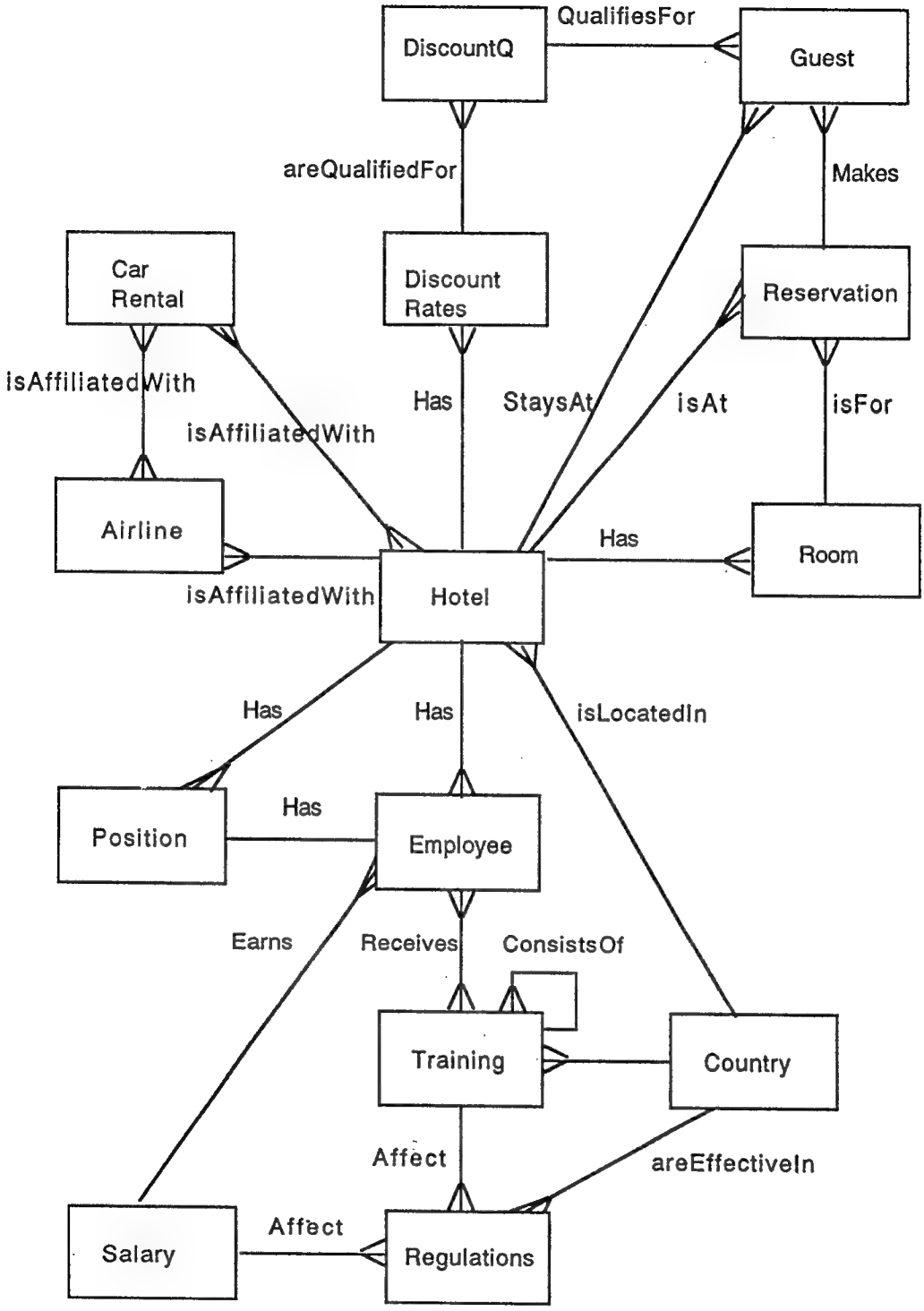

Figure 3: Hotel Diagram in E-R Sublanguage

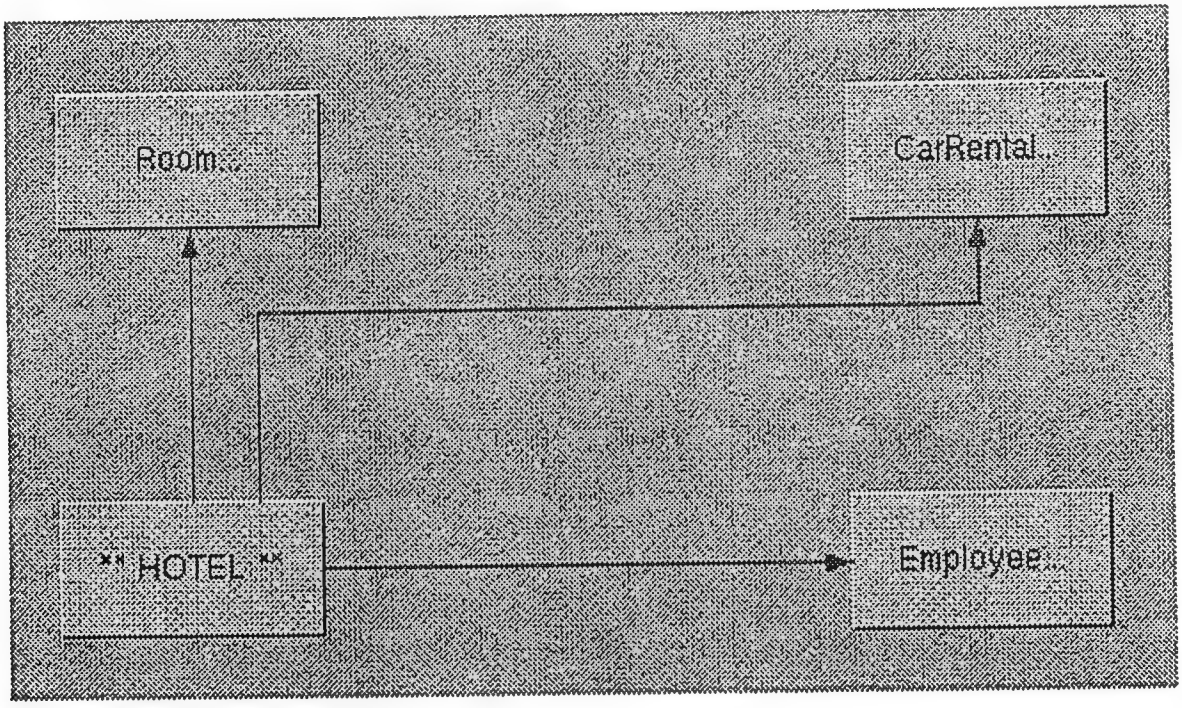

Figure 4: Hotel Diagram Overview (from demo)

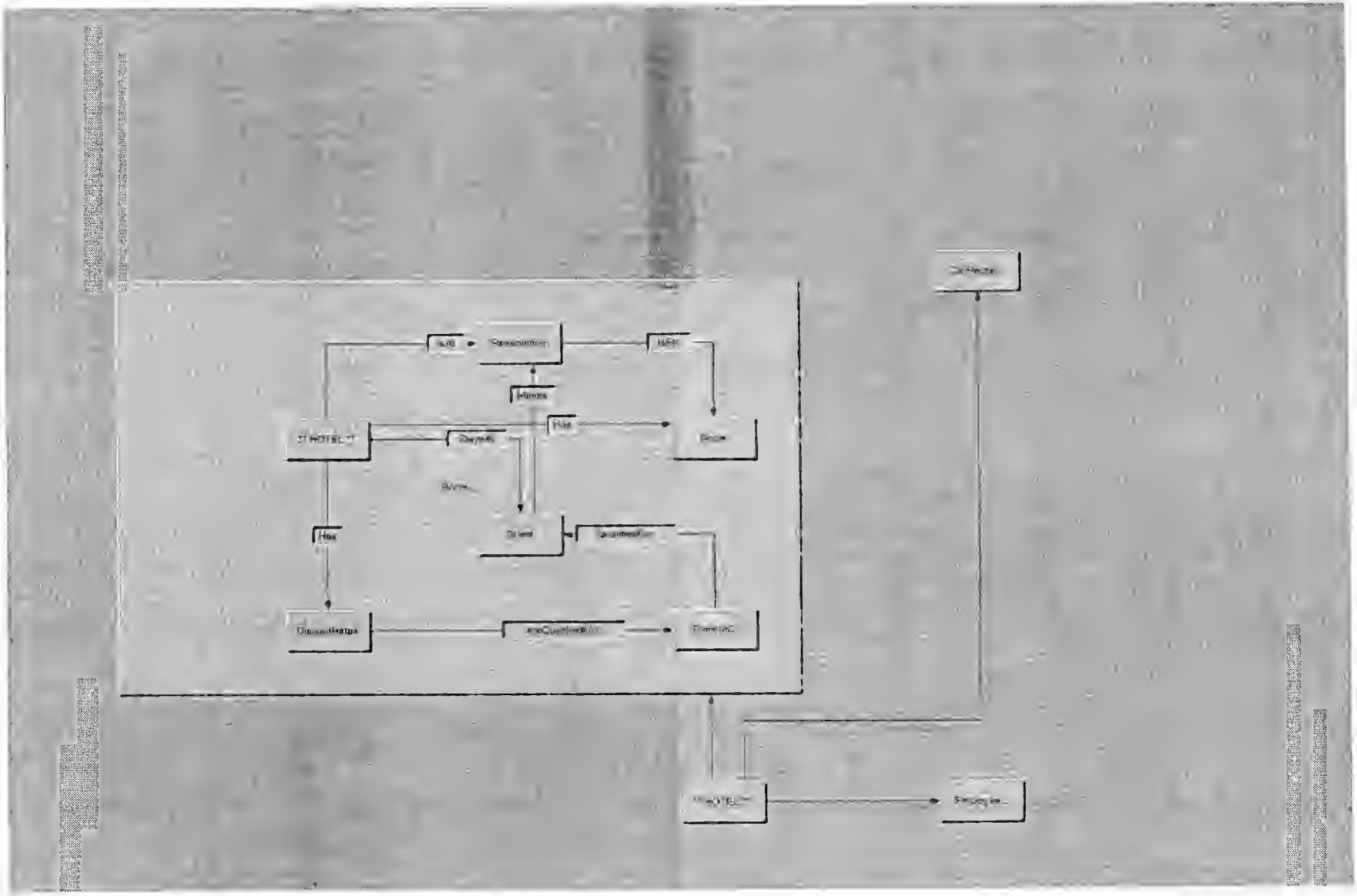

Figure 5: Hotel Diagram Overview with one Subgraph Expanded

as the 'crows foot' are used,[6] and although most edges are horizontal or vertical, some diagonal edges are also present.

Figure 4 shows an application of *diagram planning*; this is an overview diagram that shows the central Hotel node and the three subnetworks that link to it. Figure 5 shows the diagram overview with one subnetwork expanded. The diagram overview was automatically created by our demo system, using a standard clustering algorithm [BS91].

Figures 6, 7, and 8 illustrate another example of how a diagram produced by the ADM system might be improved using the techniques we are examining. Figure 6 shows a class diagram for various kinds of Documents; it was produced by the ADM AGL system.[7] Like most object-oriented class diagrams, it contains both inheritance (is-a) and relational links. Unfortunately, this makes the diagram difficult to display, because relational links should ideally be shown with a network-type layout, while inheritance links should ideally be shown with a tree or hierarchical layout. Figure 7 was produced automatically from Figure 6 by our demo system; in it, only relational links that apply to the superclass are displayed, using ADM's network layout mode. Inheritance links and relations between subclasses have been placed into a separate subdiagram, which is drawn in ADM's hierarchical mode, and is shown in Figure 8. Separating the links into different diagrams allows each type of information to be drawn according to the layout mode (sublanguage) which is most appropriate for it.

[6]The crows foot is a set of branching lines, and specifies a 'many' cardinality for a relation. For example, since Hotels can have many Rooms, there is a crows foot on the Room end of the Hotel Has Room relation.

[7]This is the Document module from the file `building-blocks.msl`, drawn in non-incremental layout mode.

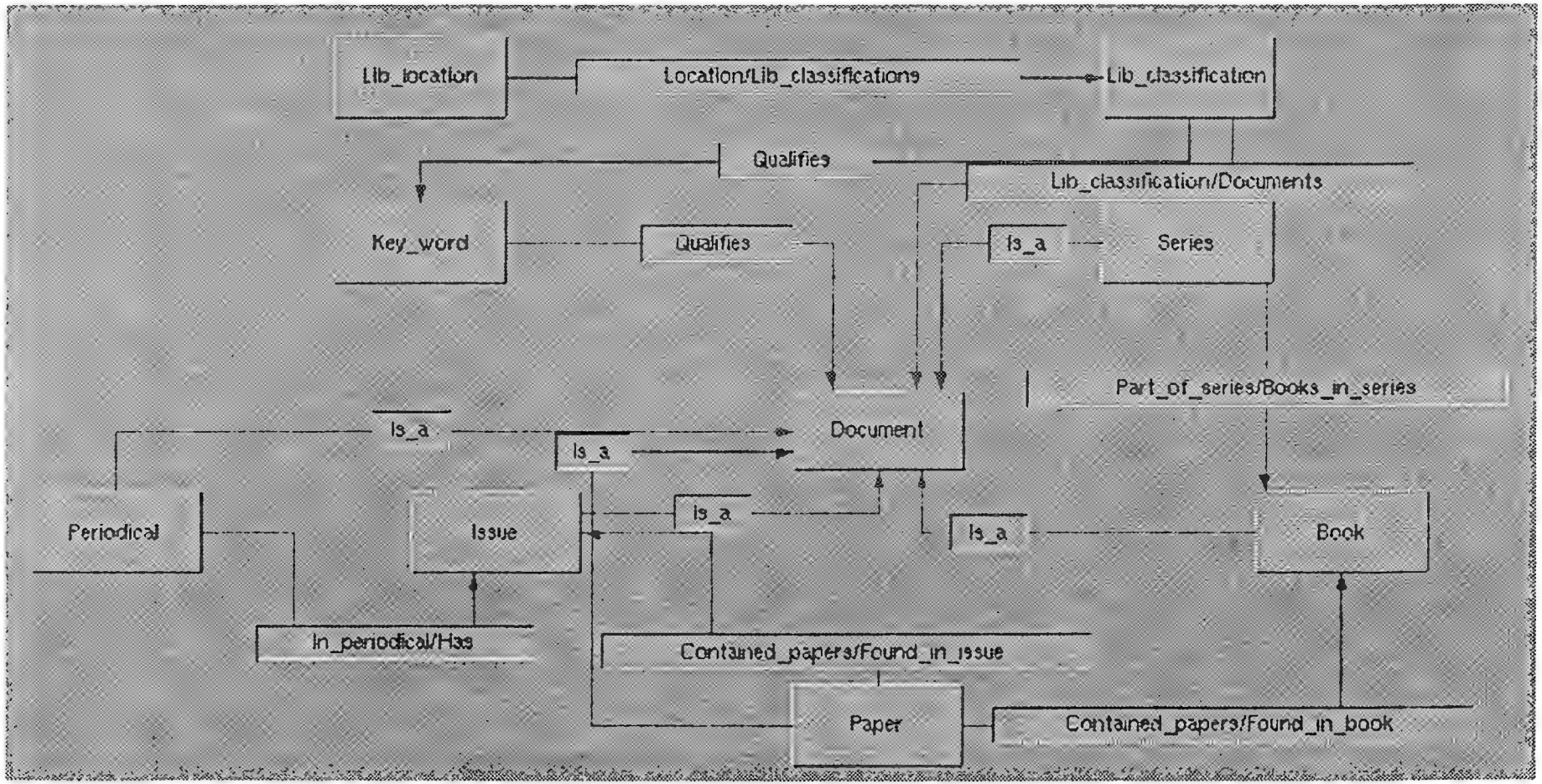

Figure 6: Document Diagram as Displayed by ADM

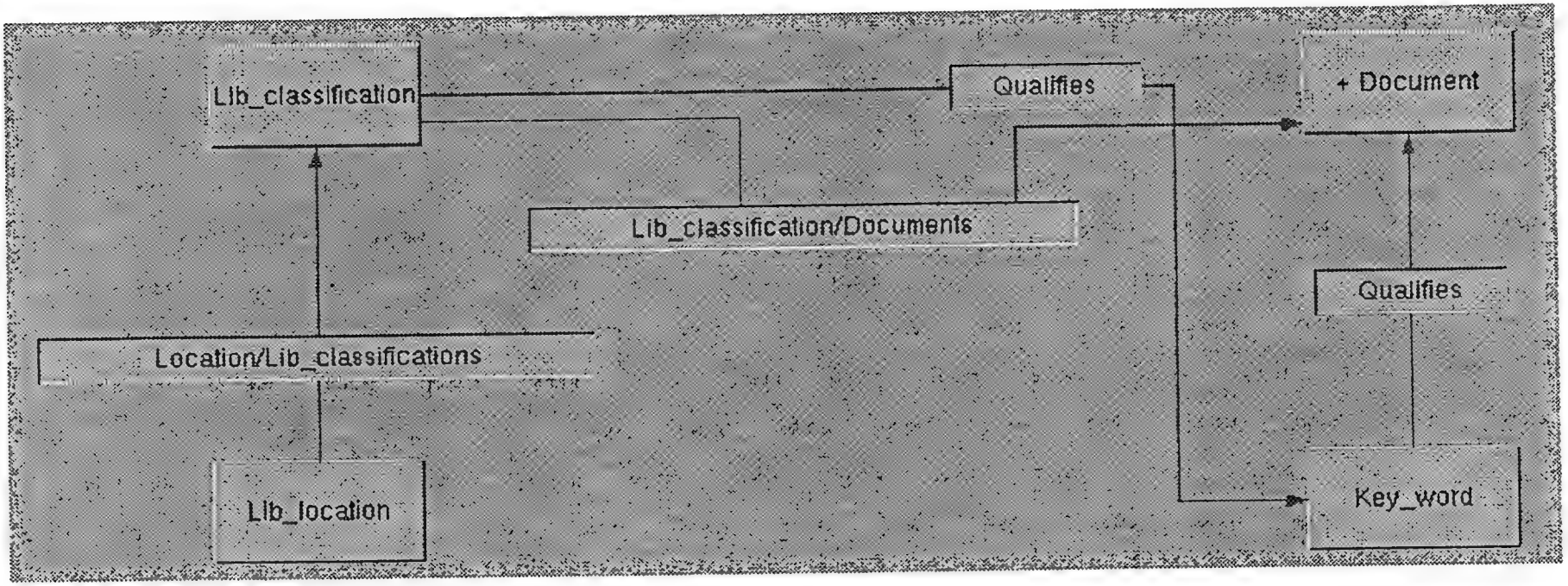

Figure 7: Document Diagram with Inheritance Information Suppressed

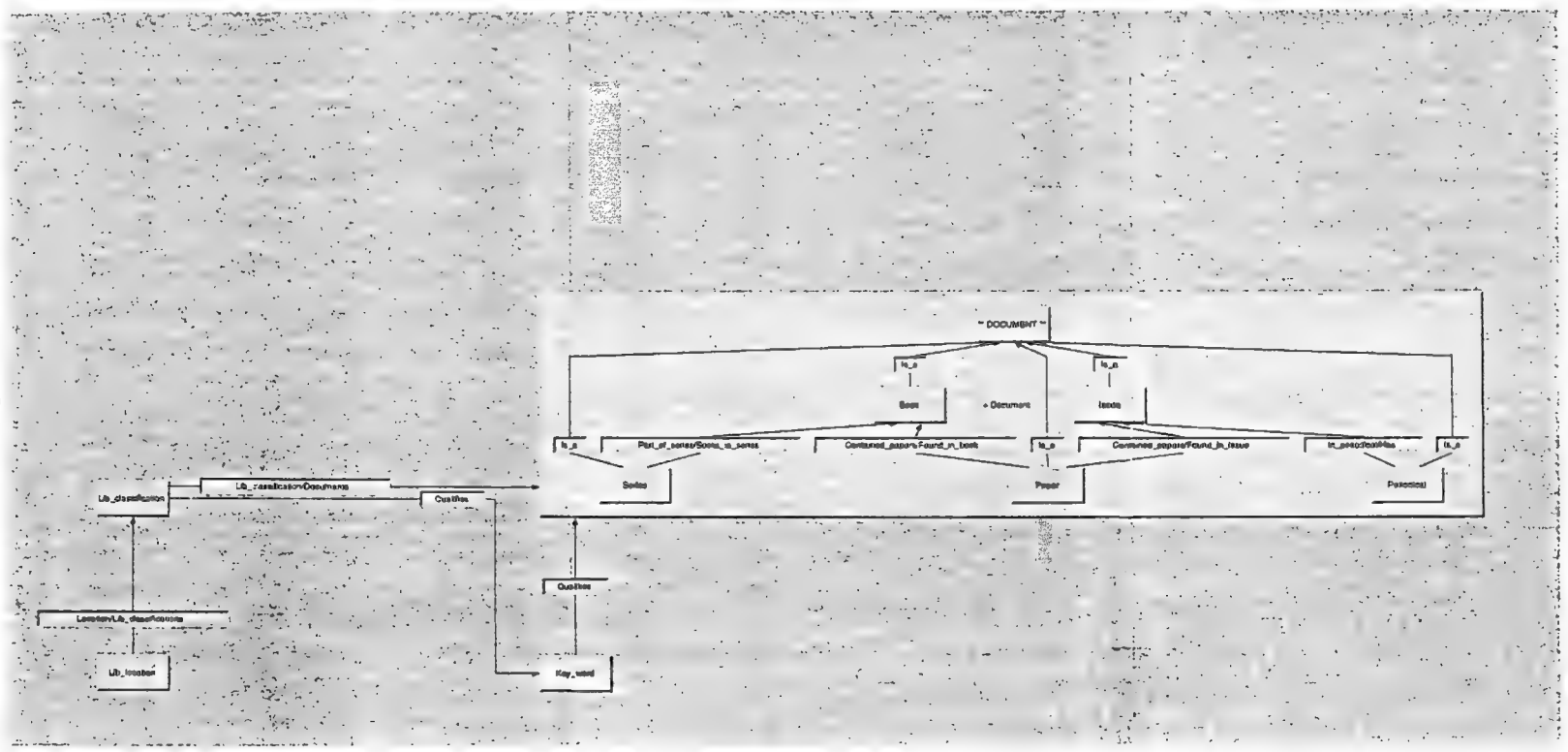

Figure 8: Inheritance Subdiagram for Document Diagram

2 Previous and Related Work

2.1 Academic Research on AGL

There is an extensive academic literature on AGL for network diagrams; hundreds of papers have been published on the topic, and there is annual conference on Graph Drawing. In very general terms, most academic AGL research has concentrated on investigating efficient algorithms, and relatively little work has been done on characterizing what criteria a diagram should satisfy in order to be understandable and effective. In other words, a 'typical' AGL research paper will assume some fairly general criteria (see below), and then propose an efficient algorithm for generating layouts that meet these constraints under certain conditions. [ET89] is an extensive AGL bibliography that focuses on algorithmic papers.

An example of a paper from this genre that is specifically targeted towards generating E-R diagrams is [TBT83]. The following diagram criteria are used:

- Minimize edge crossings
- Minimize bends in lines
- Minimize line length
- Use equal-length connections between relationship boxes and related entity boxes

The first three criteria are 'universal' rules that the paper's first author, Roberto Tamassia (a well-known figure in the AGL community), has used for many other kinds of diagrams as well; the fourth rule is only applicable for E-R diagrams. This set is representative of the kind of rules assumed by most 'algorithmic' AGL research.

2.2 Experienced Practitioner Rules

Besides the algorithm-oriented AGL work mentioned above, there is also a substantial body of articles and books by graphic designers and other practitioners on what makes diagrams and data graphics effective; [Ber83, Tuf83, Tuf90] are some well-known examples. Within the CASE field, there are also many textbooks that outline rules for drawing effective diagrams, again based

on the insights of experienced practitioners. A typical example of such a set of rules, from Richard Barker's textbook on E-R diagrams [Bar90], appears below:

- Grouping: if a group of entities form a 'functional group', they should be laid out together, in a subdiagram.
- Alignment: entity boxes should be lined up.
- Lines: relationship lines should mostly be horizontal or vertical; however, some diagonal lines should also be present. A diagram should not contain large numbers of parallel relationship lines. Line crossings should be avoided; if two lines must cross, the angle between them should ideally be between 30 and 60 degrees. Bends in lines should be avoided if possible.
- Overall shape: each diagram should have a unique 'overall shape', so that people can quickly recognize it.
- Ordering: if a relationship has a 'many' end, it should be at the left or at the top of the relation.
- Positioning: as a consequence of the ordering rule above, many highly-significant entities will be in the lower-right corner of a diagram.
- General layout: plenty of white space should be used, to avoid cluttering the diagram.

Barker's rules are similar to, but not identical with, E-R drawing rules given by other experts; James Martin [Mar87, MO92], for example, does not use Barker's ordering rule for relations with a 'many' end. At least some of Barker's rules also agree with psychological knowledge about the human visual system. For example, the rule that the angle between two lines should be at least 30 degrees agrees with findings that the human visual system responds quickly to orientation differences of 30 degrees or more [Kos94, page 267].

There obviously is a big difference between these 'experienced practitioner' diagramming rules, and the heuristics incorporated into most academic AGL systems (Section 2.1). It would be interesting to ask someone like Richard Barker or James Martin what they thought of rules such as the ones given in Section 2.1; we have not, unfortunately, been able to do so.

2.3 Algorithms

As mentioned above, the bulk of AGL research has concentrated on algorithmic issues. There have been two general approachs:

- Design an algorithm that incorporates a specific set of layout rules (e.g., [TBT83, MHT93]).
- Build a general constraint-satisfaction engine that takes as input both a specification of the diagram and a specification of the layout rules (e.g., [KMS94, GN94]).

The first approach seems to be much more popular, at least judging from the number of papers published on it. Systems built with it have in the past generally tended to be very fast but somewhat inflexible. Systems built around constraint-satisfaction approaches, in contrast, have tended to be slower but more flexible. However, in recent years some "embedded rule" systems have become more flexible by parameterizing their algorithms in various ways; the ADM AGL system is a good example of such a system. On the other hand, constraint-based systems such as [GN94] have become faster through the use of more sophisticated constraint-satisfaction algorithms. We do not in this report state a preference between the two approaches; we have focused more on the issue of 'what should AGL systems produce' instead of 'how should AGL systems work'.

2.4 Commercial Systems

AGL technology is starting to appear in commercial systems; Rational's ROSE Object-Oriented CASE system, for example, includes some AGL capabilities. We suspect this may partially be intended to support ROSE's reverse engineering capabilities; Rational stresses their ability to recover design information from legacy code, and this requires being able to automatically draw diagrams from the extracted design information.

The ADM system being developed by Andersen Consulting for Rome Laboratory includes very sophisticated AGL abilities, based on state-of-the-art algorithms. There seems to be a good chance that some of ADM's AGL technology will be transferred to industry.

AGL has also been used outside of the CASE area. Tom Sawyer's Graph Layout Toolkit [Tom94], for example, has been used by several customers to

construct diagrams of computer networks; and AT&T's dotty system [NK94] has been used to display data structures, process graphs, and other visualizations of software structures.

Not surprisingly, commercial systems tend to use the most stable and well-developed AGL algorithms. Tree-drawing algorithms (e.g., [RT81]) and the STT algorithm [STT81] for drawing hierarchical graphs (e.g., PERT charts) seem to be especially popular. Many systems also add specialized algorithms that may in practice be primarily useful in specific genres. Tom Sawyer, for example, has 'Circular' and 'Symmetric' layout styles which are very useful for computer network diagrams, but probably are inappropriate for producing CASE-related diagrams.

3 Sublanguages

3.1 Sublanguages in NLG

The concept of 'sublanguages' [GK86] is central to CoGenTex's approach to Natural-Language Generation. We strongly believe that practical NLG systems will need to produce output texts that obey numerous domain-specific constraints and conventions on syntax, word choice, text structure, etc. In other words, it is not possible to build a 'general NLG' system that can be used in any domain without customization; at best one can build a 'general NLG shell' with appropriate hooks for domain-specific customization.

3.2 Sublanguages in AGL

We believe sublanguages also exist and are very important in graphics. For example, the diagram-drawing rules given in CASE textbooks (e.g., Barker's rules for E-R diagrams, given in Section 2.2) essentially are attempts to define conventions, i.e., sublanguages, for drawing certain kinds of diagrams. On a more abstract level, the psychologist Stephen Kosslyn [Kos94], has pointed out that an experienced human reader can interpret a graphic much more quickly if the graphic uses a 'display format' that the reader is familiar with; this enables the reader to interpret the graphic by looking for visual patterns that he or she has seen in other graphics, instead of having to interpret the graphic from first principles.

In other words, it is important that diagrams be *consistent* (Section 8), i.e., display similar kinds of information in similar ways, and sublanguages are a very effective way of achieving consistency. For example, in the abstract it doesn't matter whether flowchart boxes that represent sequential operations are shown vertically aligned (i.e., top-to-bottom) or horizontally aligned (i.e., left-to-right). If a human viewer is used to seeing sequences shown top-to-bottom, however, than he or she will find a new flowchart much easier to understand if it also follows this 'convention'. And, similarly, an inheritance taxonomy may be easier to understand if, like most inheritance taxonomies, it is drawn with parents above (e.g., instead of below) their children, and with the root node in the middle of the diagram's top edge. It is a fundamental principle of human-machine communication [Shn87] that similar information in a similar context should be presented in a similar way.

The AGL research literature says very little about sublanguage issues, although the need to incorporate domain-specific conventions is sometimes acknowledged. The AGL systems we experimented with all claimed to a greater or lesser degree to be general-purpose, but in practice usually seemed tuned to particular kinds of diagrams. Brendan Madden, President of Tom Sawyer Software, agreed (in a personal communication) that a substantial amount of "specialization" was needed to get his general system to produce acceptable layouts in particular domains and genres; and that this specialization included adjusting the system to conform to pre-existing conventions in the genre.

The ADM system has some support for sublanguage variation, in that it can draw a diagram (or pieces of a diagram) in three different styles: tree, hierarchy, and network. In our demo system, we show a simple example of how this ability (with a small amount of added intelligence) can be used to increase the comprehensibility of a class diagram for an object-oriented system.

Several people have asked us whether sublanguages can be organized into an inheritance taxonomy. In such a taxonomy, for example, Barker's E-R sublanguage might be a specialization of a generic E-R sublanguage, which in itself was a specialization of a generic CASE diagram sublanguage, etc. At this point we can not say whether such a taxonomy would be useful or not; further research is needed to clarify this issue.

4 Pragmatics

4.1 Pragmatics in NLG

Human speakers often make pragmatic inferences from texts. For example, suppose Sam tells Jane *I really love two of my teachers, Mr. Jones and Ms. Smith. I also have Mr. Robinson as a teacher.* Jane is likely to infer from this statement that Sam does not like Mr. Robinson as much as Mr. Jones or Ms. Smith. Since this inference is not something that Sam explicitly stated, it is a *pragmatic* inference. Pragmatic correctness has been a factor in many NLG systems, including [Hov88, DR95].

4.2 Pragmatics in AGL

Humans make pragmatic inferences from diagrams as well as texts, and AGL systems need to ensure that their output is pragmatically correct as well as literally correct [MR90]. In particular:

- *Importance* is often pragmatically conveyed by position, e.g., a node in the center of a diagram may be assumed (by a human viewer) to be more important than a node in the lower-left corner. Which diagram positions imply high importance or salience depends on the sublanguage conventions of the relevant diagram genre.

- *Grouping* information can be pragmatically conveyed by proximity and alignment; a set of nodes that are close together and horizontally or vertically aligned are often assumed by human viewers to form a semantic (functional) group.

There are two sides to 'pragmatic correctness':

- The diagram should use pragmatic effects to convey useful information about relevant importance/salience, grouping, etc.

- The diagram should not accidentally convey incorrect and unwanted pragmatic inferences. A diagram that has this property is said to be *free of false implicatures* [MR90].

The second type of pragmatic correctness, freedom from false implicatures, is much harder to achieve automatically than the first, in part because we do not have good psychological models of human visual perception, e.g., when a particular structure will be perceived by a human viewer as a group. There is some promising work in Gestalt psychology which may eventually lead to such a model [BS93], but this has not happened yet.

Joe Marks has probably done the most work on pragmatics of network diagrams [Mar91b, KMS94]. Marks treats pragmatic layout features such as alignment and proximity as 'Visual Organizational Features' (VOFs). Marks has built several systems that accept as input diagram definitions that include VOFs as well as nodes and edges, and generate as output diagrams with the specified topology that also satisfy the pragmatic VOF constraints. Even Marks's systems, however, still have problems avoiding false implicatures, although they are very good at using pragmatics in a positive way.

The claim is sometimes made that rules such as 'minimize edge length' (Section 2.2) will *de facto* result in appropriate groupings. For example, we can argue that since members of a set of functionally-related nodes will probably have more links with other members of this set than with entities outside of the set, an AGL system that minimizes edge length is likely to place these nodes close together. There is probably some truth to this argument, at least for small diagrams with relatively low connectivity, but there are also cases where better layouts can be produced by explicitly taking grouping information into account. In the example diagrams included in Section 1.5, for example, we believe that groupings are more obvious in Figure 2, which was explicitly drawn with the goal of using pragmatics to convey grouping information, than they are in Figure 1, which was automatically produced by an AGL algorithm that used heuristics such as minimizing edge lengths.

Many of the systems we looked at, including the ADM, Tom Sawyer and AT&T systems, could generate diagrams with embedded 'groupings', 'clusters', or 'subdiagrams'. Some of Tom Sawyer's layout modes also seemed to include implicit pragmatic rules about positioning (e.g., 'when showing a computer network, try to put the main servers in the middle of the page'), although the product literature was not very clear about this.

Another issue related to pragmatics is how the information should be obtained in the first place; for example, how can the AGL system know that a particular set of entities form a group? In some cases it may be possible to obtain this information from semantic or domain knowledge, but it might

also be worth augmenting diagram-authoring environments to acquire this information directly from human authors. For example, in an ADM-like system, a human diagram creator could be given the option of selecting a set of nodes and explicitly indicating that these nodes form a pragmatic group.

5 Document Planning and Structure

5.1 Document Planning in NLG

Text documents usually consist of many individual paragraphs, put together in a coherent structured manner. A document that takes this form is usually more useful and comprehensible than a document that consists of a single multipage paragraph! Other structures are also possible, e.g., small paragraph-sized texts can be linked together as a hypertext network instead of being linearly ordered on a page [RML95].

5.2 Document Planning in AGL

Many textbooks on creating diagrams suggest that a structured set of diagrams is often better than a single 'include everything' diagram. For example, James Martin [MO92] suggests that instead of creating a single complex E-R diagram that may contain hundreds of nodes and communicate dozens of different kinds of information, it may be better to create a simple 'summary/overview' diagram that only shows the dozen or so most important nodes, and has hypertext (hypergraphics) links to diagrams that give detailed pictures of portions of the system. Similarly, some kinds of information (e.g., details on entity attributes, such as whether an attribute is optional or required) may best be shown in a separate 'attribute information' window which appears when the user clicks on the main screen, instead of in the main E-R diagram.

In other words, individual diagrams should be kept simple. Each diagram should only contain a relatively small number of nodes, and communicate just a few kinds of information. When it is necessary to describe a system that has a large number of nodes and many kinds of information, this should be done by creating many separate diagrams, each of a manageable size and complexity. If an on-line system is used, these diagrams can be linked together by hypertext mechanisms.

Within an AGL context, keeping individual diagrams simple has the added advantage that most (current) AGL algorithms do a much better job on small diagrams than on large ones. It is much easier to automatically produce an effective and comprehensible layout of a 10-node diagram than of a 100-node diagram.

Commercial drawing packages (e.g., for creating flowcharts) often allow users to connect individual diagrams via hypertext links, and suggest that this be used to link summary/overview and detailed diagrams. Advanced AGL systems such as the one in ADM allow users to specify that a set of nodes is a subgraph, which can then be collapsed so that only a single node appears for the entire subgraph; this is another way of getting an overview.

Some experimental hypertext systems use automatic clustering algorithms to decompose a large graph. Purely syntactic approaches, e.g., forming subgraphs out of biconnected components [BS91], seem most popular. Such syntactic criteria are probably less effective than semantic criteria (e.g., that a certain set of nodes all convey information about one entity), but they are also more easily implemented. In our demo system, we show that the biconnected-component algorithm, modified to be suitable for E-R diagrams, can produce plausible clusterings of at least some of the diagrams used in ADM.

The hypertext community has also examined techniques such as *fisheye views* [SB94] that simultaneously (in the same diagram) give both a diagram overview and a close-up of one portion of the diagram. Such techniques could probably be adapted to the display of CASE diagrams, but we are unsure how appropriate they would be in this context.

Some data-graphics AGL systems (e.g., SAGE [RMM91]) limit the number of different *kinds* of information communicated in a single graphic, and automatically produce several graphics if many kinds of information need to be communicated. These systems do not attempt to limit the number of individual pieces of information communicated by a single graphic.

6 User and Task Tailoring

6.1 User/Task Tailoring in NLG

There has been a sizable body of work in the NLG community on tailoring texts according to a model of the user's expertise and task. Tailoring can take place at both a low level (e.g., syntactic tailoring as presented in [BP89]), and at a high level (e.g., determining content and overall structure, as presented in [Par88]). User expertise models can be quite detailed [Rei91], or simply a choice between 'novice' and 'expert'.

6.2 User/Task Tailoring in AGL

In principle, the content of a graphic should depend on the user's task; the graphic should present the information that is relevant to the user in the context of what he or she is trying to do, with a presentation style that maximizes its usefulness in context [Kos94].

In the context of producing CASE-related diagrams, we find it useful to distinguish between

- domain-independent information about the user's perceptions and preferences, e.g., whether he or she needs large fonts or is color-blind;
- high-level domain-independent task, including in particular whether the user is creating a diagram (and just wants very routine layout tasks automated); examining a diagram produced by another human author; or examining a diagram automatically produced by information extracted from legacy code or formal specifications;
- domain-dependent task and user information, such as the fact that the user is a moderately experienced project manager who is currently writing a monthly progress report for a software development project.

The first kind of information (e.g., whether the user needs large fonts) is unquestionably important and useful. Although such factors in principle affect text generation as well as graphics generation, there has been little research on them in the NLG community, so we as NLG experts may not be able to offer any special insights as to how to exploit such information.

The second kind of information (e.g., whether the user is creating or browsing a diagram) is also important, and has a significant impact on desired functionality. For example, automatic diagram decomposition with clustering algorithms (as discussed in Section 5.2) can be very useful when browsing diagrams, especially those produced automatically from specifications or legacy code; automatic decomposition may be less useful in an authoring context, on the other hand, where the author is likely to want to explicitly specify a decomposition. Conversely, incremental layout (i.e., algorithms [MHT93] which preserve as much as possible of an existing layout when small changes are made to it) is extremely useful in authoring contexts, but less important for browsing.

The third kind of information (the specific domain task the user is performing) is also unquestionably important, but utilizing it to improve diagrams may require a substantial (perhaps even prohibitive) amount of domain knowledge. In particular, there have been several attempts to tailor data graphics (e.g., charts) in such a manner, with limited success. The SAGE system [RMM91], for example, could alter data graphics according to relatively detailed task models; but Steve Roth, SAGE's creator, told us that he found it extremely difficult in practice to acquire the detailed user and task information he needed to successfully perform this kind of tailoring. It was possible to get high-level domain-relative task information, such as "explain why there is a cost overrun"; but impossible to get more detailed information about data-presentation goals, such as "I'm interested in how Current-Cost has compared to Projected-Cost over the lifetime of the project." Roth now believes that automatically tailoring presentations to the user's task only makes sense in limited high-volume domains, where a knowledge-based system can be built to convert high-level domain goals into more specific data-presentation goals that SAGE can understand. He is instead now stressing semi-automated design. In the original SAGE system, the user specified his or her goals in an abstract declarative way, and SAGE designed an appropriate graphic based on this. In the newer SAGEBRUSH and SAGEBOOK systems [GRKM94], an initial task-independent graphic is produced which the user can modify, with automated support for filling in details; in other words, the user specifies 'task-dependent' information by directly modifying the graphic to make it appropriate, and doesn't try to declaratively express what he or she is interested in.

Steve Casner also built a data graphics system that modified its output

according to a specification of the user's task [Cas91]. Casner's system was different in detail from Roth's, but his overall conclusion was the same; the system could not in practice be used for AGL, because it was too difficult to get the task information in the format the system needed.

7 Psychological Basis

7.1 Psychology and NLG

There have been scattered efforts in the NLG community to base algorithms and heuristics on psychological knowledge about human language abilities. Scott and Souza [SS90], for example, based a text-structuring algorithm on knowledge of how human hearers comprehended texts, while Dale and Reiter [DR95] based a referring-expression generation algorithm on knowledge of how human speakers performed this task.

7.2 Psychology and AGL

To the best of our knowledge, there has been no effort to base AGL systems for CASE-related diagrams on psychological models of humans perception or creation of diagrams. In all likelihood, this is largely due to the fact that the current state of knowledge of how people create and perceive diagrams is very patchy. For example, a recent book [Kos94] by the psychologist Steve Kosslyn, one of the leaders in this field, summarized all relevant psychological findings in four pages. To be sure, this included some useful information, such as the fact that the visual system most easily detects angle differences of 30 degrees or greater. This presumably implies that AGL systems should try to ensure that non-parallel lines differ in angle by at least 30 degrees, and indeed we find something similar to this in Barker's 'experienced practitioner' rules for creating E-R diagrams (Section 2.2).

Overall, however, we probably have a long way to go before we can base a significant fraction of AGL heuristics on psychological knowledge. And this is a pity. It certain would be nice, for example, to use Gestalt psychology as a model of how people perceive pragmatic grouping phenomena (Section 4.2), which several people have suggested (e.g., [BS93]), but the models presented (at least the ones we are familiar with) are too vague and under-specified to be incorporated into a computer algorithm.

8 Consistency

8.1 Consistency in NLG

The importance of consistency in NLG depends on the genre. In many technical genres, it is important that a piece of information be presented in exactly the same way every time it occurs. This is especially true in contexts where texts are read by non-native speakers with limited fluency in English (or whatever the target language is); such people are likely to become confused if, for example, many synonyms are used in a text. Consistency is usually enforced by making authors conform to a set of writing rules (e.g., [AEC86]); such rules may, for example, include lists of allowed and illegal words, and acceptable and unacceptable grammatical constructs. On the other hand, in non-technical genres such as newspaper and literary writing, it is often preferable to vary wording and syntax, i.e., to use *different* presentations of information and *not* to be consistent.

8.2 Consistency in AGL

Consistency in graphics may be more universally important than in text. The human visual system has sophisticated pattern-recognition abilities, and diagrams will be more effective and useful if they exploit these abilities when communicating information [Kos94], by giving each 'information pattern' a distinct 'visual pattern' that the user can recognize from previous experience. This is recognized by practitioners, e.g., Barker's 'overall shape' rule for E-R diagrams (Section 2.2). The human language system may not have such a well-developed pattern-recognition ability, which may explain why consistency is less universally important in language.

There has been some work on consistency in AGL systems, e.g., [Mar91a], and we believe that sublanguages (Section 3) are probably a powerful way of enforcing consistency in graphics. It is not clear to us if there are any other NLG techniques that can be applied to automating consistency in AGL.

9 A Multimodal Document-Creation Theory

We have in this project investigated whether Natural-Language Generation (NLG) ideas can be applied to diagram generation problems. We have stressed here the practical side of this work, i.e., the potential of using this analysis to improve the effectiveness of AGL systems. But there is also a very interesting theoretical prospect, namely the prospect of a 'multimodal' theory of document creation. If we can identify common principles that lie behind the creation of both effective texts and effective graphics, this may allow us to create a sound theoretical basis for 'multimedia' systems that communicate information to humans by a combination of text and graphics, plus also perhaps techniques such as animation and virtual reality. And if we can separate 'media-dependent' and 'media-independent' principles of communication with humans, this may lead us to significant insights about the cognitive workings of the human mind, and where the line is drawn between 'general communication abilities' and 'media-specific hardware'.

We are a long ways from being able to propose even the outline of such a theory. But we find it exciting that we have identified so many similarities between 'NLG principles' and 'AGL principles'; this suggests that there really may be possible to propose a 'unified theory' of computer-to-human communication. And it is also interesting that at least one of the differences we have conjectured, namely the fact that consistency is probably more universally important in graphics than in text, may perhaps be directly traceable to differences in the human perceptual mechanism, i.e., the fact that the human visual system has greater pattern recognition and matching abilities than the human language system.

Somewhat to our surprise, by the way, there has been relatively little previous research on comparing the underlying principles behind text generation and graphics generation (some exceptions are [MR90, AR93]). There has been a substantial amount of work on multimodal systems that combine text and graphics, e.g., [MEF+90, WAF+93]. Most of this work, however, has concentrated on integration issues, such as determining what kind of information is best communicated by each modality, and coordinating generated text with generated graphics [FM90, WAGR91].

10 Conclusion

A good Automatic Graph Layout (AGL) system is very useful in an advanced CASE environment. But the quality of diagrams produced by current AGL systems is sometimes less than ideal, and this is hindering the spread and acceptance of AGL technology. Based on our expertise in Natural-Language Generation (NLG) technology, we believe that AGL systems will become more effective and produce significantly better diagrams if they:

- Accept the need to adjust layout rules to fit the conventions or *sublanguages* that have evolved in the different graphical genres.
- Use *pragmatic* features such as alignment and proximity to convey information about salience, grouping, etc.
- Use *document planning* techniques to decompose large diagrams into related sets of small diagrams.
- Take into consideration the *user's task*, at least at a high level, and ensure that the functionality they provide is appropriate and useful for the expected tasks.

In short, AGL systems should take *contextual factors* (target genre, pragmatic factors, user's task, etc.) into consideration when they draw a graph, and not just look at the literal data that needs to be presented to the diagram.

There are also some techniques which perhaps could not realistically help commercial AGL systems in the short-term, but which are worth considering for longer-term research efforts. These include

- Ensuring that diagrams do not mistakenly convey false pragmatic inferences, i.e., making sure that generated diagrams are *free of false implicatures.*
- Basing graph layout on a low-level and detailed model of exactly what task the user is undertaking.
- Basing layout heuristics on psycholinguistic knowledge of the human visual system.

Finally, we have been impressed by the numerous similarities we have found in the underlying issues and techniques for AGL and NLG. In particular, this suggests that it may be possible to create a common 'multimodal' theory of how computers should communicate information to humans, regardless of the manner in which the information is presented. Such a theory would be of great practical importance, and also of great theoretical interest, and we can only hope that our small study helps point the way towards the construction of an integrated multimodal theory.

Acknowledgements

We are very grateful to all of the people who have helped us with this project, including Elizabeth André, Steve Feiner, Winfried Graf, Robert Inder, Tanya Korelsky, Edy Liongosari, Brendan Madden, Joe Marks, Kanth Miriyala, Bhaskar Naidu, Steve North, Jon Oberlander, Thomas Rist, Nancy Roberts, Steve Roth, Keith Stenning, and Doug White. It goes without saying, of course, that we alone are responsible for the final content of this report. We are also very grateful to Rome Laboratory, who supported this work under contract F30602-94-C-0281.

References

[AEC86] AECMA. A guide for the preparation of aircraft maintenance documentation in the international aerospace maintenance language, 1986. Available from BDC Publishing Services, Slack Lane, Derby, UK.

[AR93] Elizabeth André and Thomas Rist. The design of illustrated documents as a planning task. In Mark Maybury, editor, *Intelligent Multimedia Interfaces.* AAAI Press, 1993.

[Bar90] Richard Barker. *CASE* Method: Entity Relationship Modelling.* Addison-Wesley, 1990.

[BP89] John Bateman and Cecile Paris. Phrasing a text in terms the user can understand. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence (IJCAI-1989)*, volume 2, pages 1511–1517, 1989.

[Ber83] J. Bertin. *Semiology of Graphics: Diagrams, Networks, Maps.* University of Wisconsin Press, 1983.

[BS93] Rens Bod and Remko Scha. Deriving optimal network diagrams by means of structural information theory, 1993. Chapter from Deliverable 5.1 of GRACE project. Human Communication Research Centre, University of Edinburgh, Edinburgh, Scotland.

[BS91] Rodrigo Botafogo and Ben Shneiderman. Identifying aggregates in hypertext structures. In *Proceedings of Hypertext-1991*, pages 63–74, 1991.

[CK94] David Caldwell and Tatiana Korelsky. Bilingual generation of job descriptions from quasi-conceptual forms. In *Proceedings of the Fourth Conference on Applied Natural Language Processing (ANLP-1994)*, pages 1–6, 1994.

[Cas91] Stephen Casner. A task-analytic approach to the automatic design of graphic presentations. *ACM Transactions on Graphics*, 10:111–151, 1991.

[DR95] Robert Dale and Ehud Reiter. Computational interpretations of the gricean maxims in the generation of referring expressions. *Cognitive Science*, 19(2), 1995. Forthcoming.

[DMBS92] Michael DeBellis, Kanth Miriyala, Sudin Bhat, and Bill Sasso. KBSA Concept Demo final report. Technical report, Rome Laboratory, 1992.

[ET89] P. Eades and R. Tamassia. Algorithms for drawing graphs: An annotated bibliography. Technical Report CS-89-09, Dept. of Computer Science, Brown University, 1989.

[FM90] Steve Feiner and Kathleen McKeown. Coordinating text and graphics in explanation generation. In *Proceedings of the 8th National Conference on Artificial Intelligence (AAAI-1990)*, volume 1, pages 442–449, 1990.

[GDK94] Eli Goldberg, Norbert Driedger, and Richard Kittredge. Using natural-language processing to produce weather forecasts. *IEEE Expert*, 9(2):45–53, 1994.

[GRKM94] Jade Goldstein, Stephen Roth, John Kolojejchick, and Joe Mattis. A framework for knowledge-based, interactive data exploration. *Journal of Visual Languages and Computing*, 1994. Forthcoming.

[GN94] Winfried Graf and Stefan Neurohr. Using graphical style and visibility constraints for a meaningful layout in visual programming interfaces. Research Report RR-94-15, DFKI, Saarbruecken, Germany, 1994.

[GK86] Ralph Grishman and Richard Kittredge, editors. *Analyzing Language in Restricted Domains: Sublanguage Description and Processing.* Lawrence Erlbaum, 1986.

[Hov88] Eduard Hovy. *Generating Natural Language under Pragmatic Constraints.* Lawrence Erlbaum, 1988.

[IKK+92] L. Iordanskaja, M. Kim, R. Kittredge, B. Lavoie, and A. Polguère. Generation of extended bilingual statistical reports.

In *Proceedings of the 14th International Conference on Computational Linguistics (COLING-1992)*, volume 3, pages 1019–1023, 1992.

[KMR93] Tatiana Korelsky, Daryl McCullough, and Owen Rambow. Knowledge requirements for the automatic generation of project management reports. In *Proceedings of the Eighth Knowledge-Based Software Engineering Conference (KBSE-1993)*, pages 2–9, 1993.

[KMS94] Corey Kosak, Joseph Marks, and Stuart Shieber. Automating the layout of network diagrams with specified visual organization. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(3), 1994.

[Kos94] Stephen Kosslyn. *Elements of Graphic Design.* W.H. Freeman, New York, 1994.

[Mar91a] Joe Marks. Discourse coherence and the consistent design of informational graphics. In *Proceedings of the AAAI-1991 Workshop on Intelligent Multimedia Interfaces*, pages 29–36, 1991.

[Mar91b] Joseph Marks. *Automating the Design of Network Diagrams.* PhD thesis, Harvard University, 1991.

[MR90] Joseph Marks and Ehud Reiter. Avoiding unwanted conversational implicatures in text and graphics. In *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-1900)*, pages 450–456, 1990.

[Mar87] James Martin. *Recommended Diagramming Standards for Analysts and Programmers.* Prentice-Hall, 1987.

[MO92] James Martin and James Odell. *Object-Oriented Analysis and Design.* Prentice-Hall, 1992.

[MEF+90] Kathleen McKeown, Michael Elhadad, Yumiko Fukumoto, Jong Lim, Christine Lombardi, Jacques Robin, and Frank Smadja. Natural language generation in COMET. In Robert Dale, Chris Mellish, and Michael Zock, editors, *Current Research in Natural*

Language Generation, pages 103–139. Academic Press, London, 1990.

[MHT93] Kanth Miriyala, Scot Hornick, and Roberto Tamassia. An incremental approach to aesthetic graph layout. In *Proceedings of the Sixth International Workshop on Computer-Aided Software Engineering (CASE-1993)*, pages 297–308, 1993.

[NK94] Stephen North and Eleftherious Koutsofios. Applications of graph visualization, 1994. Unpublished manuscript. AT&T Bell Laboratories, Murray Hill, NJ.

[Par88] Cecile Paris. Tailoring object descriptions to the user's level of expertise. *Computational Linguistics*, 14(3):64–78, 1988.

[RT81] Edward Reingold and John Tilford. Tidier drawing of trees. *IEEE Transactions on Software Engineering*, SE-7(2):223–228, 1981.

[Rei91] Ehud Reiter. A new model of lexical choice for nouns. *Computational Intelligence*, 7(4):240–251, 1991.

[RML95] Ehud Reiter, Chris Mellish, and John Levine. Automatic generation of technical documentation. *Applied Artificial Intelligence*, 9, 1995. Forthcoming.

[RMM91] Stephen Roth, Joe Mattis, and Xavier Mesnard. Graphics and natural language as components of automatic explanation. In Joseph Sullivan and Sherman Tyler, editors, *Intelligent User Interfaces*, pages 207–239. ACM Press (Addison-Wesley), 1991.

[SB94] Manojit Sarkar and Marc Brown. Graphical fisheye views. *Communications of the ACM*, 37(12):73–84, 1994.

[SS90] Donia Scott and Clarisse Sieckenius de Souza. Getting the message across in RST-based text generation. In Robert Dale, Chris Mellish, and Michael Zock, editors, *Current Research in Natural Language Generation*, pages 47–73. Academic Press, London, 1990.

[Sea69] J. Searle. *Speech Acts: An Essay on the Philosophy of Language.* Cambridge University Press, Cambridge, 1969.

[Shn87] Ben Shneiderman. *Designing the User Interface: Strategies for Effective Human-Computer Interaction.* Addison-Wesley, 1987.

[STT81] Kozo Sugiyama, Shojiro Tagawa, and Mitsuhiko Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-11(2):109–125; 1981.

[TBT83] R. Tamassia, C. Batini, and M. Talamo. An algorithm for automatic layout of entity relationship diagrams. In C. Davis, S. Jajodia, , P. Ng, and R. Yeh, editors, *Entity-Relationship Approach to Software Engineering (Proceedings of the Third International Conference on the Entity-Relationship Approach)*, pages 421–439. Elsevier, 1983.

[Tom94] Tom Sawyer Software. Graph layout toolkit, 1994. Available from Tom Sawyer Software, 1824B Fourth St, Berkeley, CA 94710.

[Tuf83] Edward Tufte. *The Visual Display of Quantitative Information.* Graphics Press, 1983.

[Tuf90] Edward Tufte. *Envisioning Information.* Graphics Press, 1990.

[WAF+93] Wolfgang Wahlster, Elisabeth André, Wolfgang Finkler, Hans-Jürgen Profitlich, and Thomas Rist. Plan-based integration of natural language and graphics generation. *Artificial Intelligence*, 63:387–427, 1993.

[WAGR91] Wolfgang Wahlster, Elisabeth Andre, Winfried Graf, and Thomas Rist. Designing illustrated texts: How language production is influenced by graphics generation. In *Proceedings of the Fifth Conference of the European Chapter of the Association for Computational Linguistics (EACL-1991)*, pages 8–14, 1991.

Appendix: Demo System

We have built a simple demonstration system that shows how some of our ideas might be used to improve some of the diagrams produced by the ADM AGL system. Our system functions as a pre-processor to the ADM `agl-demo` program; that is, given an existing `agl-demo` input file such as `agltest.lhotel`, it will produce a modified version of this file that can be laid out and displayed by the `agl-demo` program.

Partly because of limitations in the `agl-demo` input language, we have concentrated on diagram planning and (limited) sublanguage effects in the demo. We originally had considered incorporating pragmatic effects as well, but this proved to be difficult because we could not specify explicit node coordinates in `agl-demo` input files.

The demo's output is shown in Section 1.5. In particular,

- The demo uses the biconnected-component algorithm [BS91] to automatically decompose a complex graph into subgraphs. Figure 4 shows the components produced when this algorithm is applied to the Hotel diagram of Figure 1.

- When a diagram contains both inheritance (is-a) and relational links, the demo will separate the inheritance information into a separate subdiagram that is drawn in ADM's 'hierarchical' mode. Top-level relational links will be left in the main diagram, which is drawn in ADM's 'network' mode. This allows different types of information to be displayed in different graphical sublanguages. Figures 7 and 8 show the diagrams produced by the demo from the Document diagram shown in Figure 6.

To run the demo, you will need a Sun-4 (SPARC) workstation that has ADM installed, including the `agl-demo` program. The demo is supplied on a Sun/UNIX tar-format tape; it is installed simply by unloading the demo files onto the workstation.

To run the demo, `cd` into the demo directory and type

`processGraph` *agl-demo-input-file*

For example,

```
processGraph agltest.lhotel
```

This will produce an output file called `output.agl`. To view this file, run the `agl-demo` program, and open `output.agl` (use the `Open...` option beneath the `File` menu).

Rome Laboratory

Customer Satisfaction Survey

RL-TR-_____

Please complete this survey, and mail to RL/IMPS, 26 Electronic Pky, Griffiss AFB NY 13441-4514. Your assessment and feedback regarding this technical report will allow Rome Laboratory to have a vehicle to continuously improve our methods of research, publication, and customer satisfaction. Your assistance is greatly appreciated.
Thank You

_____

_____

Organization Name:_____(Optional)

Organization POC: _____(Optional)

Address:_____

1. On a scale of 1 to 5 how would you rate the technology developed under this research?

5-Extremely Useful 1-Not Useful/Wasteful

Rating_____

Please use the space below to comment on your rating. Please suggest improvements. Use the back of this sheet if necessary.

2. Do any specific areas of the report stand out as exceptional?

Yes___ No_____

If yes, please identify the area(s), and comment on what aspects make them "stand out."

3. Do any specific areas of the report stand out as inferior?

Yes___ No___

If yes, please identify the area(s), and comment on what aspects make them "stand out."

4. Please utilize the space below to comment on any other aspects of the report. Comments on both technical content and reporting format are desired.

*U.S. GOVERNMENT PRINTING OFFICE: 1995-610-126-50234

MISSION

OF

ROME LABORATORY

Mission. The mission of Rome Laboratory is to advance the science and technologies of command, control, communications and intelligence and to transition them into systems to meet customer needs. To achieve this, Rome Lab:

a. Conducts vigorous research, development and test programs in all applicable technologies;

b. Transitions technology to current and future systems to improve operational capability, readiness, and supportability;

c. Provides a full range of technical support to Air Force Materiel Command product centers and other Air Force organizations;

d. Promotes transfer of technology to the private sector;

e. Maintains leading edge technological expertise in the areas of surveillance, communications, command and control, intelligence, reliability science, electro-magnetic technology, photonics, signal processing, and computational science.

The thrust areas of technical competence include: Surveillance, Communications, Command and Control, Intelligence, Signal Processing, Computer Science and Technology, Electromagnetic Technology, Photonics and Reliability Sciences.